A Study on Efficient Work Stealing Based Execution of Parallel Programs for Multicore Processors

Graduate School of Systems and Information Engineering
University of Tsukuba

July 2012

ADNAN

Abstract

This thesis presents a study of work stealing based techniques of parallel programming for modern shared memory parallel multicore processors. In this thesis, the objectives of the study are to explore efficient techniques of work stealing strategy that fits on multicore and manycore as well. Work stealing by lazy task creation is a well known efficient technique for parallel task programming. Work stealing also provides load-balancing capabilitiy. Work stealing enables fine grain task scheduling efficiently so that more parallelism can be extracted from programs. The background of the research in this thesis is StackThreads/MP. StackThreads/MP is a fine-grained thread library that capable of work stealing strategy. In StackThreads/MP workers are OS threads or processors.

First, this thesis propose work stealing strategy to implement OpenMP Task. OpenMP task is a new feature of OpenMP, introduced since the OpenMP 3.0. OpenMP can be used to express both regular and irregular parallelism. However, the current OpenMP task implementation incurs both overhead and load imbalance problem. To make OpenMP task works efficiently, the workload of threads must be much larger than the overhead. In addition, Although untied task is provided, small load balancing of program cost the untied task.

Second, this thesis proposes a dynamic-length work stealing strategy in lazy-task creation technique for efficient fine-grain task scheduling. Dynamic-length refers to the dynamic number of stacked tasks. Controlling the load-balancing granularity on the basis of the number of parents in stack is one idea in this strategy so that load balancing is efficient and results in to the scalable performance.

Performance comparison on the UTS using the dynamic-length, fixed-length, and the original work stealing strategy of StackThreads/MP is performed. Comparison also is performed to the other multithreaded frameworks such as Cilk and OpenMP task im-plementations. The experiments show that the dynamic-length strategy of work stealing performs well in the irregular workload such as in UTS benchmarks, as well as in the cases of regular workload. Dynamic-length strategy works better than fixed-length strategy because it works more flexibly than the fixed-length work stealing strategy. The dynamic-length work stealing strategy can avoid load imbalance due to over stealing. It achieved efficiency up to 80% without the loss performance in other regular workload. This the-sis also proposes a parallel-loop programming construct and a reducer data structure to support the nested parallel loop that important whenever many cores available in near future. By work stealing and divide-and-conquer algorithm, implementation of nested-parallel-for has a low overhead. In addition, the reducer for the parallel loops is easy to be implemented with a small overhead. This thesis presents performance evaluation results in matrix multiplicatiopn and Sparse LU factorization with nested parallelism. With the nested parallelism in matrix multiply, 79.5% of efficiency was achieved. With nested paral-lelism in Sparse LU factorization, 41.5% of efficiency was achieved. In latter case, overhead increases in that no parallelism at the outer loop available and parallelism at inner loop is too fine-grained. Although overhead seems to be large, its parallel execution still better than the OpenMP parallel loop and cilk for as well.